

Sprint 03

Half Marathon C++

August 25, 2020



ucode

Contents



Engage	2
Investigate	3
Act: Task 00 > Template Addition	5
Act: Task 01 > Output Any	6
Act: Task 02 > Sum Of Arguments	7
Act: Task 03 > Sum File Data	8
Act: Task 04 > Output Container	9
Act: Task 05 > Square Value	10
Act: Task 06 > Is Palindrome	11
Act: Task 07 > Sort Values	12
Share	13



DESCRIPTION

Hello, friend!

Hopefully, the last few days helped you to acquire new knowledge.

Efficiency is an extremely important component for both individual qualities, and development. It takes a long time to achieve maximum results, so we suggest shortening the path to success by learning function templates and iterators. They are powerful tools in C++ that greatly simplify the job of a programmer if you use them in the complex.

For example, a function template saves a lot of time because writing a template is a one-time procedure and it can be used with different types of data. Once you get used to writing function templates, you will understand that it takes no more than writing one regular function. A function template makes it much easier to support code later and it is safer since you don't have to manually overload a function by copying the code and changing only data types when you need new data type support.

Also, the use of iterators can avoid intermediate variables, lead to shorter code, consume less memory, run faster, the code is more modular and beautiful. So, this practice provides an opportunity to improve your coding skills rapidly faster.

Remember that the main goal is no longer to do more, but rather - to do less. You will need it in the future.

JUST DO IT!

BIG IDEA

Productivity.

ESSENTIAL QUESTION

How to achieve max efficiency during C++ code development?

CHALLENGE

Create specified function templates.

Investigate



GUIDING QUESTIONS

We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find answers, ask the students around you and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- What skills are important for a successful programmer?
- What are the benefits of using function templates?
- What is the syntax of a function template?
- What are the use cases of `std::istream_iterator`?
- What are the use cases of `std::input_iterator_tag`?
- What operations does `std::input_iterator_tag` support?
- What does `std::ostream_iterator` template describe?
- What are the use cases of `std::output_iterator_tag`?
- What are the use cases of `std::forward_iterator_tag`?
- Which iterator can be used to access the sequence of elements in a range in both directions (end and beginning)?
- What are the use cases of `std::random_access_iterator_tag`?
- In what cases is it better not to use the template functions?

GUIDING ACTIVITIES

Complete the following activities. Don't forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

- Read the tasks carefully and try to find as much information as possible about them.
- Consider the algorithms found in the tasks.
- Allocate your resources and time.
- Use your research to carry out the tasks below.
- Open the story and read.
- Arrange to brainstorm tasks with other students.
- Clone your git repository that is issued on the challenge page in the LMS.
- Try to implement your thoughts in code.
- Push the solution to the repository.

ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.

- Be attentive to all statements of the story.



- Analyze all information you have collected during the preparation stages. Try to define the order of your actions.
- Perform only those tasks that are given in this document.
- Submit your files using the layout described in the story. Only useful files allowed, garbage shall not pass!
- Tasks in `shell` must be executed with `zsh`.
- Compile files with commands `cmake . -Bbuild && cmake --build ./build` that will call `CMake` and build an app.
- Pay attention to what is allowed in a certain task. Use of forbidden stuff is considered a cheat and your tasks will be failed.
- Complete tasks according to the rules specified in the [Google C++ Style Guide](#). But there are several exceptions for the guide listed below:
 - you can use `#pragma once` directive instead of `#ifndef ... #define`
 - variables can be written in `mixed case`
 - class data members must begin with `m_` prefix (m for member)
 - indent 4 spaces at a time
 - each line of text in your code must be at most 120 characters long
 - ignore the sections `Inputs and Outputs` and `Legal Notice and Author Line` in the style guide
- The solution will be checked and graded by students like you. [Peer-to-Peer learning](#).
- If you have any questions or don't understand something, ask other students or just Google it.
- In the name of Talos, use your brain!

Act: Task 00



NAME

Template Addition

DIRECTORY

t00/

SUBMIT

templateAddition.h

DESCRIPTION

Create a function template `add` with two parameters that compiles with the given `main.cpp` from the **SYNOPSIS**. If parameters are strings, it concatenates them. If parameters are numeric, it adds them the mathematically.

RETURN

Returns the result of addition of two parameters.

SYNOPSIS

```
#include <iostream>
#include <string>

#include "templateAddition.h"

int main(int argc, char** argv) {
    double d1 = 3;
    double d2 = 4.3;
    std::cout << add(d1, d2) << std::endl;

    std::string s1 = "Hello";
    std::string s2 = " there";
    std::cout << add(s1, s2) << std::endl;
    return 0;
}
```

CONSOLE OUTPUT

```
> ./templateAddition
7.3$
Hello there$
>
```

SEE ALSO

[Templates](#)

Act: Task 01



NAME

Output Any

DIRECTORY

t01/

SUBMIT

outputAny.h

DESCRIPTION

Create a function template `outputAny` that prints the contents of a container to the standard output. One item per line. The function template must work correctly with all containers except those, where the iterator points to a pair.

SYNOPSIS

```
template <class Container>
void outputAny(const Container& c);
```

Act: Task 02



NAME

Sum Of Arguments

DIRECTORY

```
t02/
```

SUBMIT

```
sumOfArguments.h
```

DESCRIPTION

Create a template function `sumOfArguments` that takes a varying number of arguments as a parameter and sums them up.

RETURN

Returns the sum of a varying number of arguments.

SYNOPSIS

```
template <class T, class ...Ts>
T sumOfArguments(T t, Ts... args);
```


Act: Task 03



NAME

Sum File Data

DIRECTORY

t03/

SUBMIT

sumFileData.h

DESCRIPTION

Create a function template that takes a file name as a parameter. The file may contain integers or doubles separated by a single space. The file contents will always be valid, no need for error handling.

Use `std::istream_iterator` for the solution.

RETURN

Returns the sum of numbers listed in the file.

SYNOPSIS

```
template <class T>
T sumFileData(const std::string& fileName);
```

CONSOLE OUTPUT

```
>cat intExample_1.txt | cat -e
1 2 3 4 5$
>./sumFileData intExample_1.txt | cat -e
15$
>cat doubleExample_1.txt | cat -e
0.1 0.2 0.3 0.4 0.5$
>./sumFileData doubleExample_1.txt | cat -e
1.5$
>cat intExample_2.txt | cat -e
0 1 2 3.1 4.2 5.3$
>./sumFileData intExample_2.txt | cat -e
6$
>cat doubleExample_2.txt | cat -e
0 1 2 3.1 4.2 5.3$
>./sumFileData doubleExample_2.txt | cat -e
15.6$
>
```

SEE ALSO

`std::istream_iterator`
Input iterator

Act: Task 04



NAME

Output Container

DIRECTORY

t04/

SUBMIT

outputContainer.h

DESCRIPTION

Create a function template `outputContainer` that prints the contents of a container to the standard output. One item per line. Still no pairs here.

Use `std::ostream_iterator` for the solution.

SYNOPSIS

```
template <typename Container>
void outputContainer(const Container& container);
```

CONSOLE OUTPUT

```
> ./outputContainer | cat -e
1$
9$
8$
6$
>
```

SEE ALSO

`std::ostream_iterator`
Output iterator

Act: Task 05



NAME

Square Value

DIRECTORY

```
t05/
```

SUBMIT

```
squareValue.h
```

DESCRIPTION

Create a function template `squareValue` that iterates over a container's contents and squares its values.

SYNOPSIS

```
template <class Container>
void squareValue(Container& container);
```

SEE ALSO

[Forward iterator](#)

Act: Task 06



NAME

Is Palindrome

DIRECTORY

t06/

SUBMIT

isPalindrome.h

DESCRIPTION

Create a function template `isPalindrome` that takes two iterators as parameters and checks if they contain a palindrome.

For example, `{ 1, 2, 3, 2, 1 }`, `{ 0.1, 0.2, 0.3, 0.2, 0.1 }`, `"radar"` are palindromes.

RETURN

Returns `true` if it is a palindrome, and `false` otherwise.

SYNOPSIS

```
template <class BidirectionalIterator>
bool isPalindrome(BidirectionalIterator begin, BidirectionalIterator end);
```

SEE ALSO

[Bidirectional iterator](#)

[Palindrome](#)

Act: Task 07



NAME

Sort Values

DIRECTORY

```
t07/
```

SUBMIT

```
sortValues.h
```

DESCRIPTION

Create a function template `sortValues` that sorts the iterator values in ascending order. The usage of standard functions for sorting is **FORBIDDEN**, you must implement a sorting algorithm by yourself.

RETURN

Returns an iterator to the beginning of a sorted range.

SYNOPSIS

```
template <class RandomAccessIterator>
RandomAccessIterator sortValues(RandomAccessIterator begin, RandomAccessIterator end);
```

SEE ALSO

[Random-access iterator](#)

Share



PUBLISHING

Last but not least, the final stage of your work is to publish it. This allows you to share your challenges, solutions, and reflections with local and global audiences. During this stage, you will discover ways of getting external evaluation and feedback on your work. As a result, you will get the most out of the challenge, and get a better understanding of both your achievements and missteps.

To share your work, you can create:

- a text post, as a summary of your reflection
- charts, infographics or other ways to visualize your information
- a video, either of your work, or a reflection video
- an audio podcast. Record a story about your experience
- a photo report with a small post

Helpful tools:

- [Canva](#) - a good way to visualize your data
- [QuickTime](#) - an easy way to capture your screen, record video or audio

Examples of ways to share your experience:

- [Facebook](#) - create and share a post that will inspire your friends
- [YouTube](#) - upload an exciting video
- [GitHub](#) - share and describe your solution
- [Telegraph](#) - create a post that you can easily share on Telegram
- [Instagram](#) - share photos and stories from ucode. Don't forget to tag us :)

Share what you've learned and accomplished with your local community and the world. Use [#ucode](#) and [#CBLWorld](#) on social media.